Multiagent Flight Control in Dynamic Environments with Cooperative Coevolutionary Algorithms

Mitchell Colby
Oregon State University
colbym@engr.oregonstate.edu

Matt Knudson
NASA Ames Research Center
matt.knudson@nasa.gov

Kagan Tumer
Oregon State University
kagan.tumer@oregonstate.edu

Abstract

Dynamic environments in which objectives and environmental features change with respect to time pose a difficult problem with regards to planning optimal paths through these environments. Path planning methods are typically computationally expensive, and are often difficult to implement in real time if system objectives are changed. This computational problem is compounded when multiple agents are present in the system, as the state and action space grows exponentially with the number of agents in the system. In this work, we use cooperative coevolutionary algorithms in order to develop policies which control agent motion in a dynamic multiagent unmanned aerial system environment such that goals and perceptions change, while ensuring safety constraints are not violated. Rather than replanning new paths when the environment changes, we develop a policy which can map the new environmental features to a trajectory for the agent while ensuring safe and reliable operation, while providing 92% of the theoretically optimal performance.

Introduction

Dynamic environments where agents are constrained occur often in real-world problems such as air traffic coordination and car coordination. Mobile robot coverage tasks such as those involving Unmanned Aircraft Systems (UAS) are continuously becoming more prevalent in industrial, military, and academic applications, in part due to their fast deployment times and ability to reach areas that ground locomotive systems cannot reach [Caballero et al., 2008]. One important area of research is payload directed flight, where UAS must obtain as much information from an area as possible in a given amount of time, and potentially change their flight plans based on dynamic information obtained from the environment. A simple example of payload directed flight is UAS monitoring a forest fire. As the behavior of the fire changes, the UAS may need to alter its flight plan in order to continually provide as much information about the fire as possible. Payload directed flight is an interesting testbed for multiagent learning algorithms because the environment and system objectives change with respect to time (dynamic), and there are safety requirements which must be satisfied (constraints).

Traditional search algorithms are capable of developing optimal flight plans, but respond poorly to highly dynamic environments such as those found in payload directed flight. Further, planning algorithms are often are too computationally expensive to make route adjustments in real time for multiagent systems with high congestion and agent coupling, especially as the number of agents or size of the environment grows.

In order to address the challenge of and further the ability to dynamically adjust routes in a multiagent payload directed flight system, this research incorporates cooperative coevolutionary algorithms. Control policies which change flight trajectories based on dynamic environmental data are learned, allowing for real time trajectory adjustments based on changes in the state space. Such control policies become necessary as search algorithms become too slow to operate in real-time. In these missions, flight safety is extremely important; safety requirements, such as minimum separation between aircraft, must always be met. We must ensure that safety violations will not occur when learned policies (rather than planning) are utilized. The contributions of this work are to demonstrate that:

- a cooperative coevolutionary algorithm results in a multiagent policy such that no safety constraints are violated in any converged solution found.
- a cooperative coevolutionary algorithm outperforms finite time horizon deterministic search algorithms in multiagent payload directed flight (achieving 92% of the theoretically optimal solution).

The rest of this paper is organized as follows. Section 2 covers work related to this research. Section 3 gives details on the domain and algorithm used in this research. Section 4 details the experiments and results from this research. Finally, Section 5 discusses these results and presents future avenues of research.

Background

The following sections give detail on work related to this research, including the domain of payload directed flight, search algorithms, cooperative coevolutionary algorithms, and the use of difference evaluation functions.

Payload Directed Flight

The aim of Payload Directed Flight (PDF) is to provide guidance, navigation, and control for flight vehicles to achieve mission goals related to the payload sensors while ensuring various constraints are satisfied [Lee, Yeh, and Ippolito, 2010]. Research in PDF typically focuses on trajectory generation, autonomous feature detection, and autopilot design concepts. These topics have been investigated in [Ippolito, 2009; Lee and Ippolito, 2009; Lee, Yeh, and Ippolito, 2010]. As goals and objectives in PDF missions become more complex or the physical size of the area to be investigated grows larger, multiagent PDF missions may become necessary. Little research has been conducted on multiagent PDF, because adding a multiagent framework makes the problem of planning trajectories in PDF much more complex due to the increased size of the action and state space.

Search Algorithms

The following sections detail recursive best first search as well as multiagent A^* search.

Recursive Best First Search Recursive Best First Search (RBFS) is a simple tree search algorithm which is similar to the recursive depth first search algorithm, except that an upper bound is kept in order to allow for better paths to be explored rather than continuing indefinitely down a single path. This upper bound is set to be the f-value of the best alternative path from the ancestor to the current node. The current node is expanded and the children nodes are each expanded. If all the child nodes exceed the upper bound then the f-value of the current node is set to the best child node's f-value. If one or more of the child nodes be less than the upper bound, then the node with the smallest f-value is visited and the upper bound is set to the lowest alternative [Russell and Norvig, 2009]. Due to the large size of the state space and dynamic nature of multiagent payload directed flight, RBFS often becomes computationally intractable, requiring finite time horizon searches producing severely suboptimal multiagent plans.

Multiagent A* Search Multiagent A*, or M*, is a variant of the A* search algorithm for multiagent domains [Wagner and Choset, 2011]. Initially, each agent performs a standard A* search to find its locally optimal path. M* assumes that there is low coupling between agents, which is true when agents are well separated in the workspace. If two agents happen to be coupled after performing their individual A* algorithms, then replanning occurs while considering both agents simultaneously. M* is proven to be complete and capable of finding minimal cost paths [Wagner and Choset, 2011]. However, the assumption that agents are not strongly coupled often fails in multiagent payload directed flight, and many paths considering the joint actions of multiple agents must be considered. In a highly congested system, the M* algorithm will become too computationally expensive to perform. Ultimately, any search algorithm becomes computationally intractable in a large multiagent PDF system, because of the large state space, extremely dynamic environment, and high coupling between agents.

Cooperative Coevolutionary Algorithms

Evolutionary algorithms (EAs) are a class of stochastic search algorithms, which have been shown to work well in domains where gradient information is not readily available [Fogel, 1994]. EAs act on an initial set of candidate solutions in order to generate new solutions and to retain solutions which show improvement. *Coevolution*, an extension of evolutionary algorithms, is well-suited for multiagent domains [Ficici, Melnik, and Pollack, 2005]. In a coevolutionary algorithm, the fitness of an agent depends on the interactions it has with other agents. Thus, assessing the fitness of each agent is context-sensitive and subjective [Panait, Luke, and Wiegand, 2006]. We focus on *Cooperative* Coevolutionary Algorithms (CCEAs), where a group of agents succeed or fail as a team [Potter and Jong, 1995]. CCEAs tend to favor stable, rather than optimal, solutions [Panait, Luke, and Wiegand, 2006], because agents in each population adapt to each other, rather than adapting to find an optimal policy [Panait, 2010].Further, as agents succeed or fail as a team, the fitness of each agent becomes context-dependent and subjective [Wiegand, Jong, and Liles, 2002]. For example, an agent may take an optimal action, but receive a poor fitness assignment because its collaborators took severely suboptimal actions. The Difference Evaluation Function aims to address the credit assignment problem found in multiagent systems.

Difference Evaluation Functions

The agent-specific Difference Evaluation Function $D_i(z)$ is given by [Agogino and Tumer, 2008]:

$$D_i(z) = G(z) - G(z_{-i} + c_i) \tag{1}$$

where $G(z)$ is the global evaluation function, and $G(z_{-i} + c_i)$ is the global evaluation function without the effects of agent i. The term c_i is the *counterfactual*, which is used to replace agent i, and must not depend on the actions of agent i. Intuitively, the Difference Evaluation Function gives the impact of agent i on the global evaluation function, because the second term removes the portions of the global evaluation function not dependent on agent i. Note that:

$$\frac{\partial D_i(z)}{\partial a_i} = \frac{\partial G(z)}{\partial a_i} \tag{2}$$

where a_i is agent i. Thus, an agent i acting to increase the value of $D_i(z)$ will also act to increase the value of $G(z)$. This property is termed *factoredness* [Agogino and Tumer, 2008].

The second term in Equation 2 removes the effects of all agents other than agent i, meaning that the Difference Evaluation Function is a strong function of agent i. This results in the Difference Evaluation Function providing a feedback

signal with much less noise than the global evaluation function $G(z)$. This property is termed *learnability* [Agogino and Tumer, 2008].

Approach

The following sections describe the domain used in this research, the types of control policies utilized, as well as the approach taken to solve the problem of multiagent PDF.

Domain

We model multiagent payload directed flight to provide a testbed for our cooperative coevolutionary algorithm. Payload directed flight is an interesting domain for multiagent research, because it has dynamic environmental features and objectives, and has constraints which must be satisfied by each agent. These characteristics are common in a wide variety of real-world problems, so domains such as multiagent payload directed flight give insight that simpler domains cannot provide.

Locations of Points of Interest (POIs) are arranged in a grid. During each episode, each POI is "active" for one fourth of the episode, meaning that the POIs to be observed change as the experiment progresses. Each POI may be observed only once.

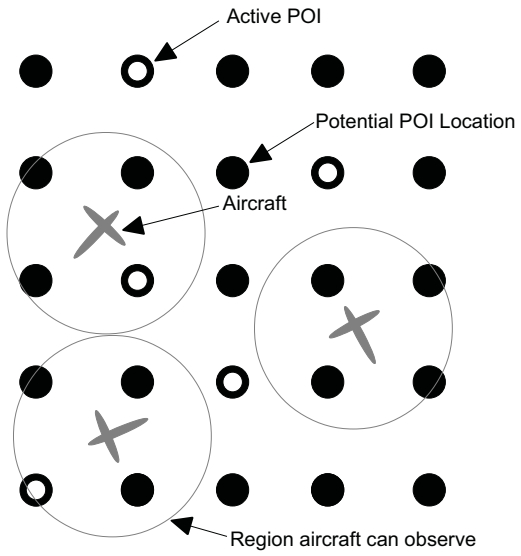

Figure 1: Payload Directed Flight domain. At any given moment, only a subset of the POIs are available to be observed. Aircraft have a finite region around them in which observations can be made.

A key concern in a multiagent PDF system is that of flight safety. Aircraft must maintain a minimum separation distance from other aircraft, in order to avoid any potential collisions, which would result in costly equipment loss and minimal observation data. In order to promote safe flight, a penalty is assessed whenever planes are within one unit of distance from each other. The system evaluation function is given as:

$$G(z) = \sum_{p \in P} n_p - n_v \beta \tag{3}$$

where n_p is an indicator function which returns 1.0 if the p'th POI was observed, and 0.0 otherwise. The variable n_v is the number of separation violations which occurred during the simulation, and β is the penalty assessed for each violation.

Control Policies

Two types of control policies are considered in this research. First, we analyze a control policy developed using RBFS. Second, we analyze a control policy utilizing neural networks.

Recursive Best First Search The first control policy analyzed is developed using RBFS. In this case, we assume each agent begins the experiment at a potential POI location (see Figure 1), and can move to any of the 8 adjacent potential POI locations at each timestep. RBFS is used to find the set of joint actions for the set of aircraft to maximize POI coverage while ensuring safety violations do not occur. Due to the exponential growth of the search space in multiagent systems, the search is limited to 5 timesteps ahead.

Neural Network Control The second control policy we use involves training neural network controllers with cooperative coevolutionary algorithms. We use neural networks because they are well-suited to handle the continuous state and action space of multiagent payload directed flight. In the case of neuro-control, planes are allowed to move continuously throughout the environment, in contrast to the discrete control policy provided by RBFS. The neural network inputs relate to a distance weighted density of planes and active POIs in each quadrant relative to the planes orientation, as in Figure 2. The distance metric is the squared Euclidian norm, bounded by a minimum observation distance of 0.2 distance units. As seen in Figure 2, the environment is split into four quadrants relative to the aircraft's orientation, denoted Q1 through Q4. The aircraft has two sensors in each quadrant. The first sensor in each quadrant q returns the sum of the inverse of the squared distance from each active POI to the aircraft (See [Colby and Tumer, 2012] for more detailed description of state variables). The second sensor in each quadrant q returns the sum of square distances from the plane to all other planes in that quadrant. The neural networks used in the research incorporate sigmoid activation functions, so network outputs are bounded between 0 and 1. The mapping from network outputs to the plane motions is found by mapping from the network output (0 to 1) to plane motion (-Δ to Δ).

Algorithm

We use a cooperative coevolutionary algorithm to train neural networks to control each agent in the multiagent PDF domain. Each agent is controlled by a single neural network,

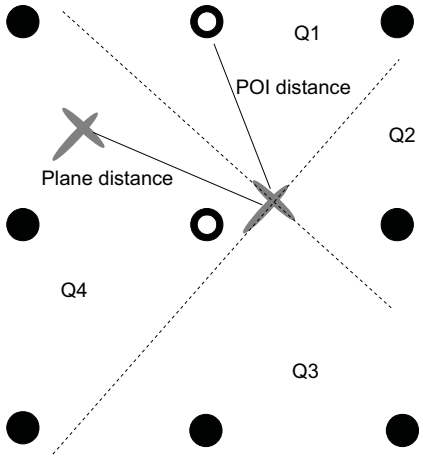

Figure 2: State inputs for aircraft. The domain is divided into four quadrants relative to the aircraft's orientation. A distance weighted density of aircraft and active POIs in each quadrant to create 8 total state inputs.

where network inputs are the relative x and y locations of the active POI locations, and the network has two outputs corresponding to the planes x and y movements.

To ensure that the system does not converge to solutions which violate safety constraints, we take two steps. First, the penalty for a safety violation is greater than the total potential maximum observation value. For example, if POIs over the course of an experiment can provide a cumulative value of 100, then penalties for safety violations are set to be greater than 100. Thus, a safety violation will always outweigh any value gained from observations. Secondly, if the algorithm begins to converge to a solution which violates safety constraints, the mutation operator increases in magnitude, pushing the solutions away from that area of the solution space. Thus, the fitness function is designed to highly discourage safety violations, and the algorithm is not allowed to converge to a solution in which safety violations occur. The CCEA used in this research is a modification of that found in [Colby and Tumer, 2012], and is given in Algorithm 1.

Algorithm 1 Cooperative Coevolutionary Algorithm (See Section for parameters)

```
Initialize N populations of k neural networks
for all Generations do
  for all Populations do
    produce k successor solutions
    mutate successor solutions
  end for
  for i = 1 → 2k do
    randomly select one agent from each population
    add agents to team T_i
    simulate T_i in domain
    assign fitness to each agent in T_i using D_j(z)
  end for
  for all Populations do
    select k networks using ε-greedy
  end for
  if suboptimalConvergence() = true then
    mutationRate = m_high
  else
    mutationRate = m_standard
  end if
end for
```

Experiments and Results

Three experiments were conducted in this research. The first experiment involves a static domain, in which each POI is activated at the beginning of the experiment and remain active for the duration of the experiment. This experiment is to provide a performance baseline in a simple domain. The second experiment uses a dynamic domain, where only a set of POIs is active at any point in time. This experiment is to demonstrate the difference between search algorithms and learning algorithms in dynamic domains where real-time flight rerouting is necessary. Finally, the third experiment involves a large domain with a large number of agents. This experiment is to demonstrate the scalability of the learning algorithm in domains which are too computationally complex for search algorithms to be conducted.

Static Domain

For the first experiment, the environment is kept static, meaning that POIs do not activate or deactivate during the experiment. A grid of 5 by 5 POIs is initialized, with 5 agents being initialized at random locations in the domain. This domain is used as a test case to demonstrate how well the cooperative coevolutionary algorithm performs in ideal conditions, as well as to demonstrate that in simple domains (small physical area, low number of agents, no dynamic changes in environment), search algorithms give optimal flight plans. A comparison of performance between RBFS and the CCEA is shown in Figure 3.

For the RBFS, once the locations of all agents are initialized, the RBFS algorithm was used to determine a flight plan such that each POI was observed, while no safety violations occur. As this particular experiment involves a small number of agents with a relatively short time window, and the environment does not change with respect to time, a full search for complete flight plans could be executed. As expected, RBFS performs optimally, creating a flight plan in which all POIs are observed with no safety violations.

After 125 statistical runs, the average performance of the CCEA corresponded to $90.12 \pm 0.54\%$ POI coverage, with a maximum of 100% coverage and a minimum of 84% coverage. However, no set of solutions violated any safety requirements in any of the 125 statistical runs. So, although the CCEA does not provide the same level of performance

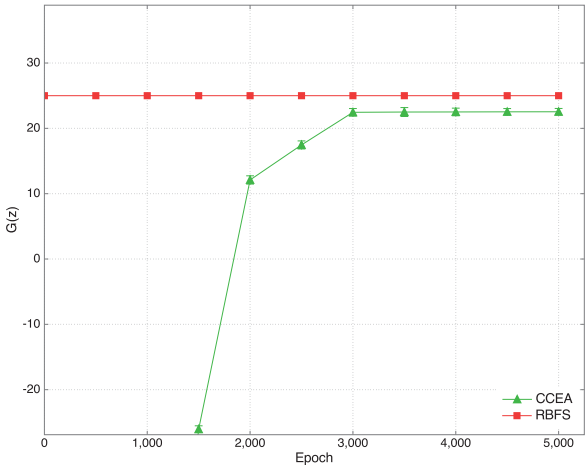

Figure 3: 5 by 5 domain with 5 agents. RBFS provides an optimal solution, while the CCEA provides an average of 90% coverage, with no separation violations.

as an optimal search algorithm, it does ensure that converged solutions are free of policies which violate safety requirements.

Dynamic Domain

For the next set of experiments, the environment is dynamic, as defined in Section . For these experiments, the potential POI locations were distributed in a 10 by 10 grid, with 5 agents. Each event in the simulation lasted for 25 timesteps, and 25 different POIs were active for each event. This domain is dynamic, as opposed to the static domain in the first experiment. Further, this domain is larger in terms of the number of POIs as compared to the first experiment. A comparison of the optimal soluton, the RBFS search, and the CCEA policies is shown in Figure 4.

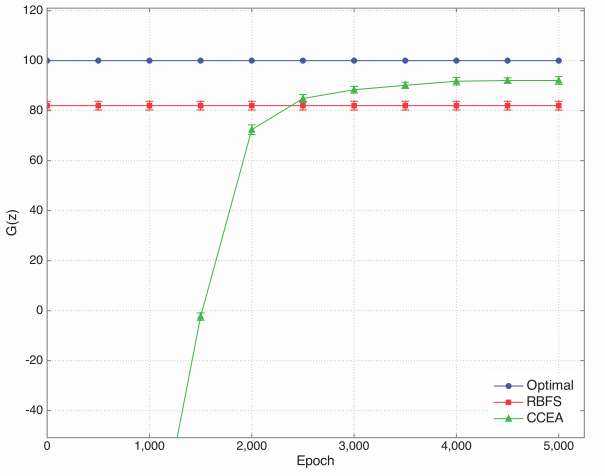

Figure 4: 10 by 10 grid with 5 agents. Finite time horizon RBFS results in 82% coverage, while the CCEA results in 92% coverage with no separation violations.

Due to the increased complexity of this domain resulting from the dynamic environment and increased number of POIs, a full RBFS search cannot be completed to create flight plans for an entire simulation. As the active POI locations change with respect to time, flight plans need to be dynamically changed in order to ensure that newly activated POIs are observed. Thus, the RBFS algorithm was carried out with a finite time window, where new flight plans were generated every 5 time steps. Every 5 time steps, an RBFS algorithm is completed to maximize the number of POIs to be observed over that time window, while ensuring that no safety violations occurred. Although the flight plans are always optimal for the finite time window, they do not form a globally optimal solution for the length of the entire simulation. This is seen in Figure 4, where the RBFS obtained an average coverage of 82% of the POIs in the domain.

For the CCEA, there were 125 statistical runs conducted, with the error bars in Figure 4 reporting the error in the mean. As in Figure 3, the error bars are often obscured by the plot symbols. After 125 statistical runs, the average performance of the CCEA corresponded to $92.05 \pm 1.56\%$ POI coverage, with a maximum of 100% coverage and a minimum of 88% coverage. Every single statistical run of the CCEA outperformed the average RBFS performance.

Large Dynamic Domain

For the final experiment, the POIs are arranged in a 15 by 15 grid, where 100 agents move throughout the environment observing POIs. As with the second experiment, the environment is dynamic, where in each of the 4 events, 56 (57 in the last event) POIs were active. This domain is too large for a search algorithm such as RBFS to be conducted. At each time step, 100^9 joint actions exist, rendering any search algorithm computationally intractable. These experiments serve to demonstrate how well the CCEA performs in a domain where search algorithms are intractable. Further, this is an exceptionally congested domain, in which separation violations are difficult to prevent. So, this experiment gives insight on how well the CCEA will perform in highly congested domains which computationally intractable for search algorithms. Results for these experiments are shown in Figure 5. As in the first two experiments, 125 statistical runs were conducted, and error in the mean results in error bars which are typically obscured by the plot symbols.

As seen in Figure 5, the CCEA performs exceptionally well, providing good coverage and no safety violations. It is of note that in this particular domain, the key difficulty is in ensuring that separation violations do not occur. The domain is extremely congested, so ensuring planes are spread out in order to prevent separation violations results in fairly good area coverage. Analysis of the statistical results show that average coverage was $92.0 \pm 1.27\%$, with a maximum coverage of 100% and a minimum coverage of 87%. This demonstrates that the CCEA is scalable in the number of agents, unlike search algorithms. It is of note that in each of the statistical runs, the converged policies lead to zero separation violations. The minimum coverage of 87% is quite high, but is obviously not optimal. This is in large part due to the penalty term associated with separation violations. As

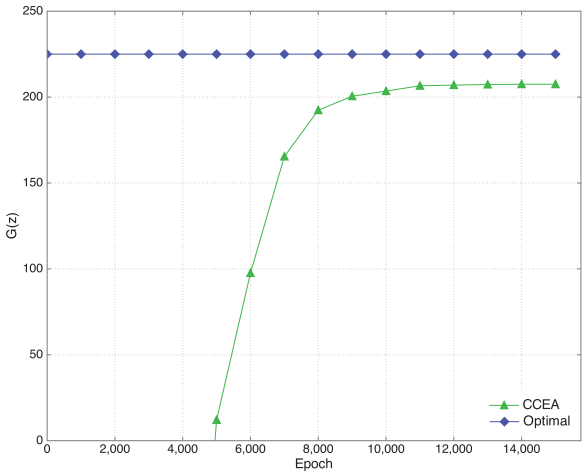

Figure 5: 15 by 15 grid with 100 agents. Due to the complexity of the domain, search algorithms are computationally intractable. The CCEA provides solutions which average 92% coverage, with no separation violations.

a separation violation negates all observation values, agents act to ensure that separation violations do not occur at the expense of POI observations. Although this results in suboptimal system performance, it ensures that separation violations do not occur. It is of note that in a hardware system, sacrificing a small amount of system performance is acceptable if safety requirements are ensured.

Discussion

Dynamic environments where agents are constrained occur often in real-world problems such as payload directed flight, air traffic coordination, and car coordination. Optimal paths may be determined, but due to the dynamic nature of the environments real-time planning will often be necessary to ensure continued high performance. As the number of the agents in the system grows, the state and action space grows exponentially, rendering traditional search algorithms intractable. Multiagent learning provides a solution which can develop control policies in these highly dynamic and constrained environments when search algorithms become too computationally expensive to implement.

This research provides three key research contributions. First, we demonstrate that cooperative coevolutionary algorithms can converge to policies such that no safety constraints are violated in multiagent payload directed flight systems. Second, we demonstrate that cooperative coevolutionary algorithms perform on average at 92% of the optimal policy. Finally, we demonstrate that cooperative coevolutionary algorithms outperform finite time horizon search algorithms, and perform well in large domains in which no search algorithms are computationally tractable at all.

Future work on this research has multiple avenues. First, we will investigate if we can improve coverage values while maintaining separation guarantees. Second, we will investigate if fitness functions other than the difference evaluation function can provide better coverage performance. Third, we will investigate heterogeneous aircraft with different payloads, and thus different objectives.

Acknowledgements

This work was completed at NASA AMES Research Center.

References

Agogino, A., and Tumer, K. 2008. Regulating Air Traffic Flow with Coupled Agents. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*.

Caballero, F.; Merino, L.; Ferruz, J.; and Ollero, A. 2008. Vision-Based Odometry and SLAM for Medium and High Altitude Flying UAVs. *Journal of Intelligent Robotic Systems* 54(1-3):137–161.

Colby, M., and Tumer, K. 2012. Shaping fitness functions for coevolving cooperative multiagent systems. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems*, 425–432.

Ficici, S.; Melnik, O.; and Pollack, J. 2005. A Game-Theoretic and Dynamical-Systems Analysis of Selection Methods in Coevolution. *Evolutionary Computation, IEEE Transactions on* 9(6):580 – 602.

Fogel, D. 1994. An Introduction to Simulated Evolutionary Optimization. *Neural Networks, IEEE Transactions on* 5(1):3 –14.

Ippolito, C. 2009. A Trajectory Generation Approach for Payload Directed Flight. *AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*.

Lee, R., and Ippolito, C. 2009. A Perception and Mapping Approach for Plume Detection in Payload Directed Flight. *AIAA Infotech, Aerospace*.

Lee, R.; Yeh, Y.; and Ippolito, C. 2010. Geological Applications of Payload Directed Flight. *AIAA Infotech, Aerospace*.

Panait, L.; Luke, S.; and Wiegand, R. 2006. Biasing Coevolutionary Search for Optimal Multiagent Behaviors. *Evolutionary Computation, IEEE Transactions on* 10(6):629 – 645.

Panait, L. 2010. Theoretical Convergence Guarantees for Cooperative Coevolutionary Algorithms. *Evolutionary Computation* 18(4):581–615.

Potter, M., and Jong, K. D. 1995. Evolving Neural Networks with Collaborative Species. In *In Proceedings of the 1995 Summer Computer Simulation Conference*, 340–345.

Russell, S., and Norvig, P. 2009. *Artificial Intelligence, a Modern Approach*. New York, USA: Prentice Hall.

Wagner, G., and Choset, H. 2011. M*: A Complete Multirobot Path Planning Algorithm with Performance Bounds. *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Wiegand, R.; Jong, K. D.; and Liles, W. 2002. Modeling Variation in Cooperative Coevolution Using Evolutionary Game Theory. In *Proceedings of the Seventh Workshop on Foundations of Genetic Algorithm*.